1-1-2002

Inverse parametric sequence alignment

Fangting Sun
Iowa State University

Inverse parametric sequence alignment

by

Fangting Sun

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:
David Fernández-Baca, Major Professor
Srinivas Aluru
Soma Chaudhuri
Govindarasu Manimaran

Iowa State University

Ames, Iowa

2002

Graduate College
Iowa State University

This is to certify that the master's thesis of

Fangting Sun

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy

TABLE OF CONTENTS

Abstract

We consider the inverse parametric sequence alignment problem, where a sequence alignment is given and the task is to determine parameter values such that the given alignment is optimal at that parameter setting. We describe a $O(mn \log n)$-time algorithm for inverse global alignment without gap penalties and a $O(mn \log m)$ time algorithm for global alignment with gap penalties, where m, n $(n \leq m)$ are the lengths of input strings. Finally, we discuss the local alignment problem and future work.

1. Introduction

Sequence Alignment

High similarity between biomolecular sequences(DNA, RNA, or amino acid sequences), usually implies significant functional or structural similarity. Thus sequence alignment, which can be used for determining similarity between biological sequences, has become essential in modern molecular biology. There are hundreds of papers written on this topic and its applications to biology. The review [1] gives relevant references.

Given two sequences S and T of lengths n and m, $n \leq m$, a *global alignment* is obtained by inserting special *space* characters into the two sequences in such a way as to build sequences S' and T' of equal length, denoted by $\mathcal{A} = (S', T')$. A *match* is a position where S' and T' have the same characters. A *mismatch* is a position in which S' and T' have different characters, neither of which is a space. An *indel* is a position in which one of S' and T' has a space. A *gap* is a sequence of one or more consecutive spaces in S' and T'.

As an example of a global alignment, consider the alignment of the strings $AGCTA$ and $ACTCA$ shown below:

A G - C T A

A C T C - A

In this alignment, character G is mismatched with C, A's and C match their counterparts in the opposite string, and all other characters are opposite space, which are called *indels.*

An global alignment $\mathcal{A}$ can be characterized by its number of matches, mismatches, indels and gaps, denoted by w, x, y, z, respectively. In scoring an alignment matches are rewarded, while mismatches, indels and gaps are penalized. Let α, β and γ denote the mismatch, indel

and gap penalties. Then the *score* of $\mathcal{A}$ is

$$score_{\mathcal{A}} = w - \alpha x - \beta y - \gamma z$$

The case, where the weight of the matches is a parameter, is ignored since we can divide all the parameters by this value and reduce it to the above case.

For any given pair of strings, there are different alignments with different scores. Continuing the previous example, consider the following two alignments for the two strings:

A G - C T A	A G C T A
A C T C - A	A C T C A

Suppose $\alpha = 1, \beta = 1, \gamma = 0$, then $score_{\mathcal{A}_1} = 3 - 1 - 2 = 0, score_{\mathcal{A}_2} = 2 - 3 = -1$, so $\mathcal{A}_1$ is better than $\mathcal{A}_2$. If $\alpha = 1, \beta = 1, \gamma = 1$, then $score_{\mathcal{A}_1} = 3-1-2-2 = -2, score_{\mathcal{A}_2} = 2-3 = -1$, so $\mathcal{A}_2$ is better than $\mathcal{A}_1$. The *optimal alignment problem* is to find a maximum-score alignment $\mathcal{A}$ between two strings. For fixed weights, this problem can be solved in $O(mn)$ time [11].

In many applications, two strings may not be highly similar in their entirety but may contain regions that are highly similar. Thus, local similarity is far more meaningful than global similarity. Then we need to find and extract a pair of regions, one from each of the two given strings, that exhibit high similarity. This is called *local alignment.*

Given two sequences S and T, a *local alignment* is obtained by finding substrings S' and T' of S and T, respectively, whose optimal global alignment score is maximum over all pairs of substrings from S and T [6].

For example, to strings $S = pqraxabcstvq$ and $T = xyaxbacsll$, given $\alpha = 1, \beta = 0.5, \gamma = 0$, then the two substrings $S' = axabcs$, $T' = axbacs$ of S and T respectively, have the following optimal alignment

a x a b - c s

a x - b a c s

which has a score of 4. Furthermore, over all choices of pairs of substrings, one from each of the two strings, those two substrings have maximum similarity. Hence, for that scoring scheme, the optimal local alignment of S and T has score 4 and is defined by substrings S' and T'.

If the lengths of S and T are n and m, then the local alignment problem can be solved in $O(mn)$ time for fixed weights of matches, mismatches, indels and gaps [12].

Parametric Sequence Alignment

When using sequence alignment methods to study sequences, there is often considerable disagreement about how to weigh matches, mismatches, indels and gaps. It is widely observed that the biological significance of the resulting alignment can be greatly affected by the choice of parameter settings. *Parametric sequence alignment* is a tool that efficiently explores such penalty variation. It avoids the problem of choosing fixed parameter settings by computing the optimal alignment as a function of variable parameters for penalties. The *parametric sequence alignment* problem is to compute optimal alignments for two fixed sequences as a function of varying penalties. The value of an alignment is a linear function of the parameters; thus the parameter space can be partitioned into *optimal regions* such that in every region one alignment is optimal throughout and the regions are maximal for this property. So *parametric alignment* allows one to see explicitly, and completely, the effect of parameter choices on the optimal alignment.

For example, consider strings $S = pqraxabcstvq$ and $T = xyaxbacsll$. Suppose $\beta = 1$, then the score of an alignment is a function of α, γ: $score = w - \alpha x - y - \gamma z$. Figure 1.1 illustrates the polygonal decomposition of the α, γ parameter space.

Parametric sequence alignment was first proposed by Fitch and Smith [4]. Later, both mathematical formulations and algorithms for *parametric sequence alignment* were obtained by Gusfield et al. [7, 8]. Figure 1.1 is drawn by the XPARAL [8], which is a parametric alignment application developed by Gusfield et al. Additional work is found in [3, 13, 14, 15].

Inverse Parametric Sequence Alignment

In *inverse parametric optimization* [2] one is given a parametric optimization problem and a desired optimal solution and the task is to determine parameter settings such that the given solution is optimal for those values. The *inverse parametric sequence alignment* problem is to

find parameter values such that reference alignment is optimal for those values or, if no such settings exist, find a parameter setting minimizing the numerical difference between the score of the optimal alignment and the score of the reference alignment. These parameter values define an *inverse optimal point* on the parameter space. Inverse parametric computation is useful for deducing parameter settings where the optimal alignment is likely to reconstruct correct alignments that have been determined by other methods.

Since the optimal regions are bounded by the intersection of hyperplanes, all regions are convex polygons [4, 7, 8, 3]. Hence, the inverse optimal parameter setting(s) must occur at a single vertex (intersection point of three or more optimal regions), at a single edge (intersection line between two optimal regions), or at a single complete polygon of the polygonal decomposition of the parameter space.

To illustrate this, let us see a simple case where only one parameter is counted. Suppose $\beta = a, \gamma = b$, where a, b are constants, then the score of alignment is the function of α: $score = w - \alpha x - ay - bz$. As shown in Figure 1.2, there are three different cases for this situation. The straight line represents the score of the reference alignment, which goes down while the value of α increases. The continuous line segments represent the score of the optimal alignments for different values of α. The score of the optimal alignments decreases when the value of α goes up. Since for different values of α there are different optimal alignments, the slopes of those line segments are different. One line segment corresponds to one complete optimal region in the decomposition of parametric space. Figure 1.2(a) shows the situation where the inverse optimal parameter settings occur at a single complete optimal region, Figure 1.2(b) shows the situation where the inverse optimal parameter setting occurs at a single vertex, Figure 1.2(c) shows the situation where the reference alignment cannot be an optimal alignment.

When more than one parameter are counted, the figure will be extended into multi-dimension in a similar but more complicated way.

Related Work

Although there has been considerable work on parametric sequence alignment, the inverse parametric sequence alignment was only mentioned in [8].

One way to locate the inverse-optimal parameter settings is to first construct the entire decomposition of the parameter space and then choose the correct values. Alternatively, one can try to find the parameter settings directly. This can be done by gradient descent [8], although it is not clear how to obtain bounds on the worst-case performance of this method. Megiddo's method of *parametric search* [9, 10] can be used instead, leading to a $O(m^2n^2)$ method for the case where only one parameter is varied. While powerful, Megiddo's method has the drawback that it leads to complex algorithms. Improvements in the running time are possible by relying on the existence of a *parallel* algorithm for the problem, but this only complicates the results further. Here we give an approach that is much simpler than Megiddo's method and exploits the integer nature of the scoring of sequence alignments.

The main idea of our algorithms is to find the inverse-optimal point in the parameter space using binary search. Our main contribution is a proof that this simple algorithm converges quickly.

The rest of this thesis is organized as follows. Chapter 2 gives a $O(mn \log n)$ algorithm for global alignment without gap penalties. A $O(mn \log m)$ algorithm for global alignment with gap penalties is described in Chapter 3. Further results are discussed in Chapter 4.

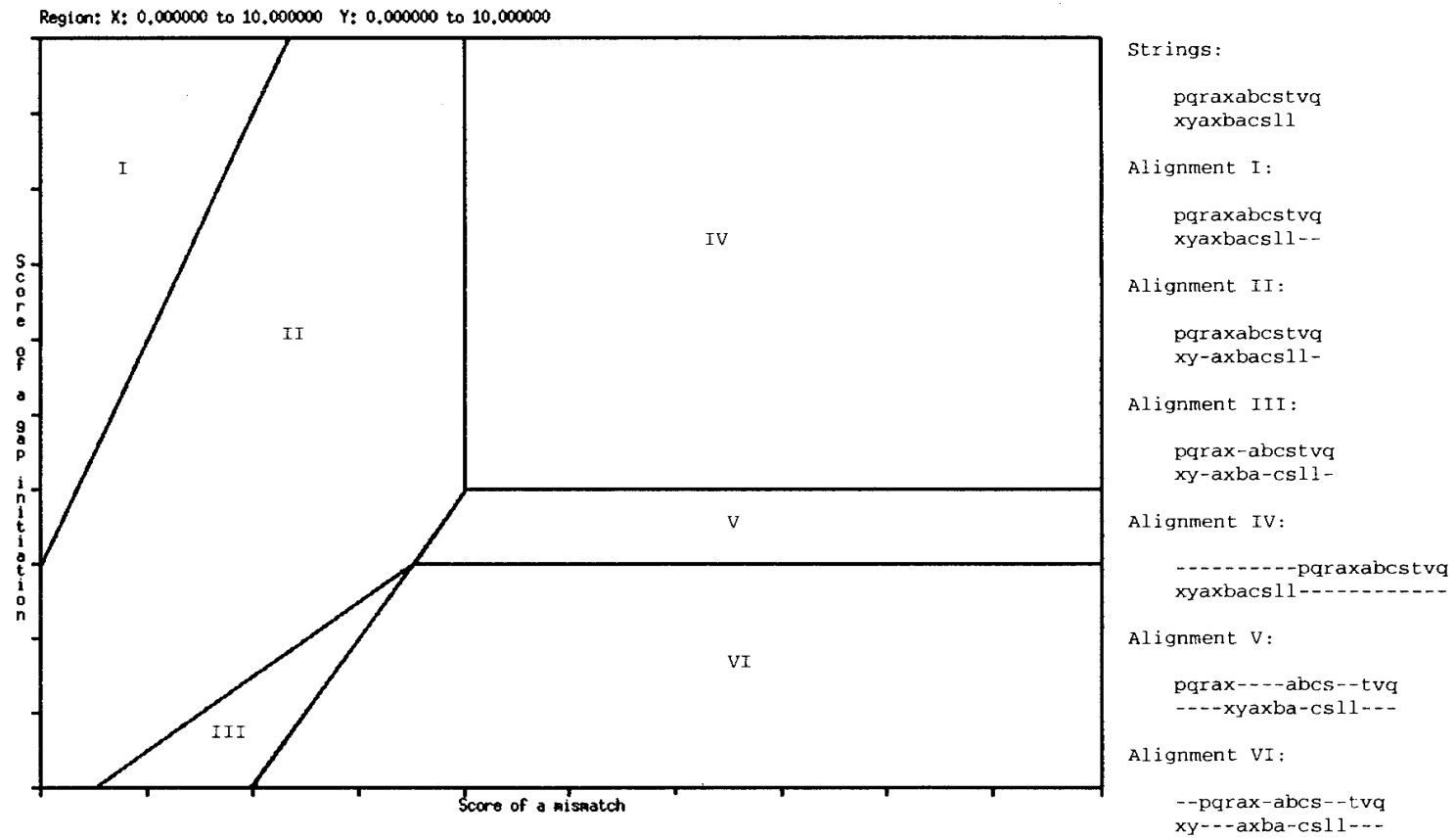

Figure 1.1: A polygonal decomposition of the α, γ parameter space

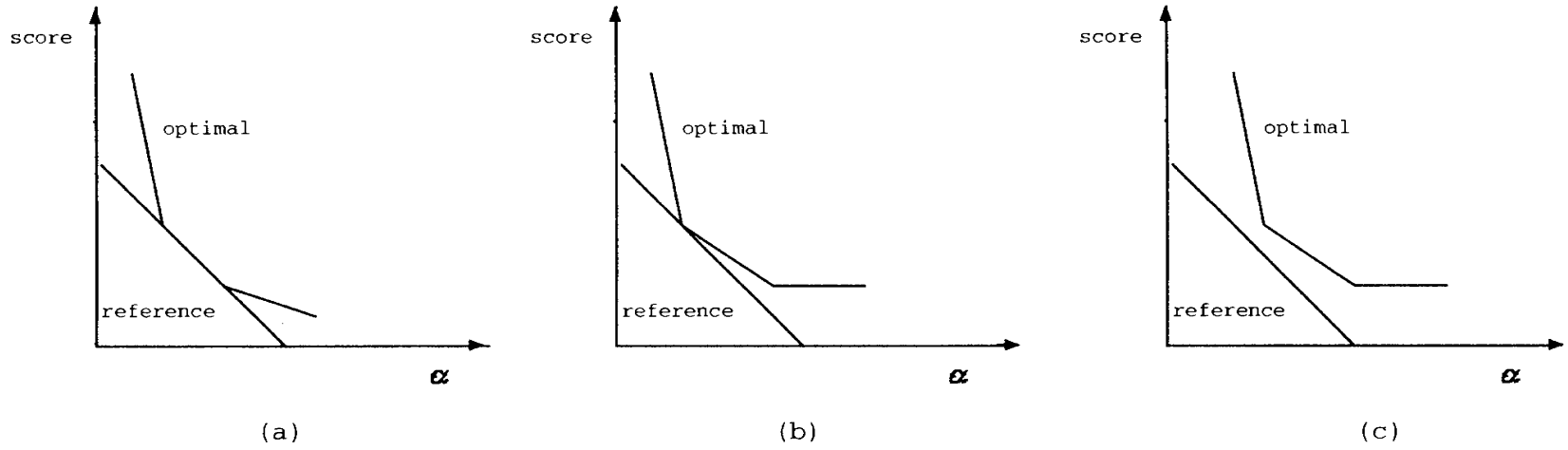

Figure 1.2: The relationship between optimal alignments and reference alignment

2. Global Alignment without Gap Penalty

In this Chapter we consider global alignment where the number of gaps is ignored ($\gamma = 0$). Then, the score function is $score = w - \alpha x - \beta y$. Given a reference alignment $\mathcal{A}_0$ with w_0 matches, x_0 mismatches and y_0 indels, we need to find an inverse-optimal point for $\mathcal{A}_0$ in the α, β plane.

Preliminary Results

Theorem 2.1 (Gusfield et al. [7]) *Any line forming a boundary between two regions is of the form $\beta = c + (c + 0.5)\alpha$, for some $c > -1/2$.*

Corollary 2.1 *Suppose (α_0, β_0) is an inverse-optimal point for reference alignment $\mathcal{A}_0$ in the α, β space. Then all points on the line that goes through $(-1, -1/2)$ and (α_0, β_0) are inverse-optimal for $\mathcal{A}_0$.*

Lemma 2.1 ([7]) *The positive β-axis intersects all the region boundaries.*

Lemma 2.2 ([7]) *Let $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$ be the optimal alignments encountered by β axis in order of increasing β-value. Then $y_{i+1} < y_i$ for all $\mathcal{A}_i (i < k)$.*

An example is shown in Figure 2.1 to illustrate the decomposition properties on the α, β parameter space when gap penalties are ignored ($\gamma = 0$).

Let $\mathcal{A}_i, \mathcal{A}_j$ be the optimal alignments in two neighboring optimal regions encountered by β axis, with score $w_i - \alpha x_i - \beta y_i$ and $w_j - \alpha x_j - \beta y_j$. Then the equation of the boundary line between the regions is:

$$\beta = \frac{w_i - w_j}{y_i - y_j} + \frac{x_j - x_i}{y_i - y_j}\alpha \tag{2.1}$$

A *breakpoint* along any given line is the point where the line moves between two adjacent optimal regions.

Lemma 2.3 *The length of the interval between any two successive breakpoints along the β-axis is greater than $1/n^2$.*

Proof According to Equation (2.1), the boundary line of two neighboring optimal regions where $\mathcal{A}_i, \mathcal{A}_j$ are optimal respectively intersects the β axis at $(0, \frac{w_i - w_j}{y_i - y_j})$.

Let $\mathcal{A}_i, \mathcal{A}_j, \mathcal{A}_k$ $(i < j < k)$ be the optimal alignments in three consecutive optimal regions when going along the β axis and let $\Delta w_1 = w_j - w_k, \Delta w_2 = w_i - w_j, \Delta y_1 = y_j - y_k, \Delta y_2 = y_i - y_j$. Then the interval between two breakpoints on the β axis is:

$$\Delta\beta = \frac{w_j - w_k}{y_j - y_k} - \frac{w_i - w_j}{y_i - y_j} = \frac{\Delta w_1 \Delta y_2 - \Delta w_2 \Delta y_1}{\Delta y_1 \Delta y_2}$$

Since $\Delta w_1, \Delta w_2, \Delta y_1, \Delta y_2$ are all integers and $\Delta\beta > 0$, then $\Delta w_1 \Delta y_2 - \Delta w_2 \Delta y_1 \geq 1$. Notice that $m - n \leq y \leq m + n$, thus $\Delta y_1 + \Delta y_2 = y_i - y_k \leq (m + n) - (m - n) \leq 2n$, therefore $\Delta y_1 \Delta y_2 \leq ((\Delta y_1 + \Delta y_2)/2)^2 \leq n^2$. It follows that $\Delta\beta \geqslant 1/n^2$. □

Algorithm and Analysis

The main idea of our algorithm is to use binary search on the β axis. The details are given in Algorithm 1.

Theorem 2.2 *Algorithm 1 correctly solves the inverse parametric alignment problem for global alignment without gaps in $O(mn \log n)$ time.*

Proof From equation (2.1), all breakpoints on the β axis lie below $(0, n)$, so it is correct to restrict the search space to the portion on the β-axis between $(0, 0)$ to $(0, n)$. Lemma 2.1 and 2.2 guarantee that binary search works for this problem, since the algorithm can decide to go up or down along the β axis according the number of indels.

If the algorithm finds a point $(0, \beta_0)$ such that the optimal alignment $\mathcal{A}$ for that point has the same number of indels as the reference alignment, then $(0, \beta_0)$ is either inverse-optimal

($w = w_0$) or it is approximately inverse-optimal ($w \neq w_0$). Following Corollary 2.1, the algorithm returns a line.

Lemma 2.3 shows that when the length of the search interval is at most $1/n^2$, it cannot contain a complete optimal region. It includes either part of one optimal region ($\mathcal{A}_{high}$ equals $\mathcal{A}_{low}$) or one breakpoint. In the first case, all points in the remaining search space are approximately inverse-optimal. In the second case, the breakpoint may be inverse-optimal or all points in the remaining search space are approximately inverse-optimal. Thus Algorithm 1 gives the correct answer.

Steps 1, 2, 4, 6-14, 17 need $O(1)$ time; step 16 and 18-28 need $O(mn)$ time; step 5 needs $O(mn)$ time, and the while statement can loop at most $3 \log n$ times. Therefore, the total time is $3 \log n \cdot O(mn) = O(mn \log n)$. □

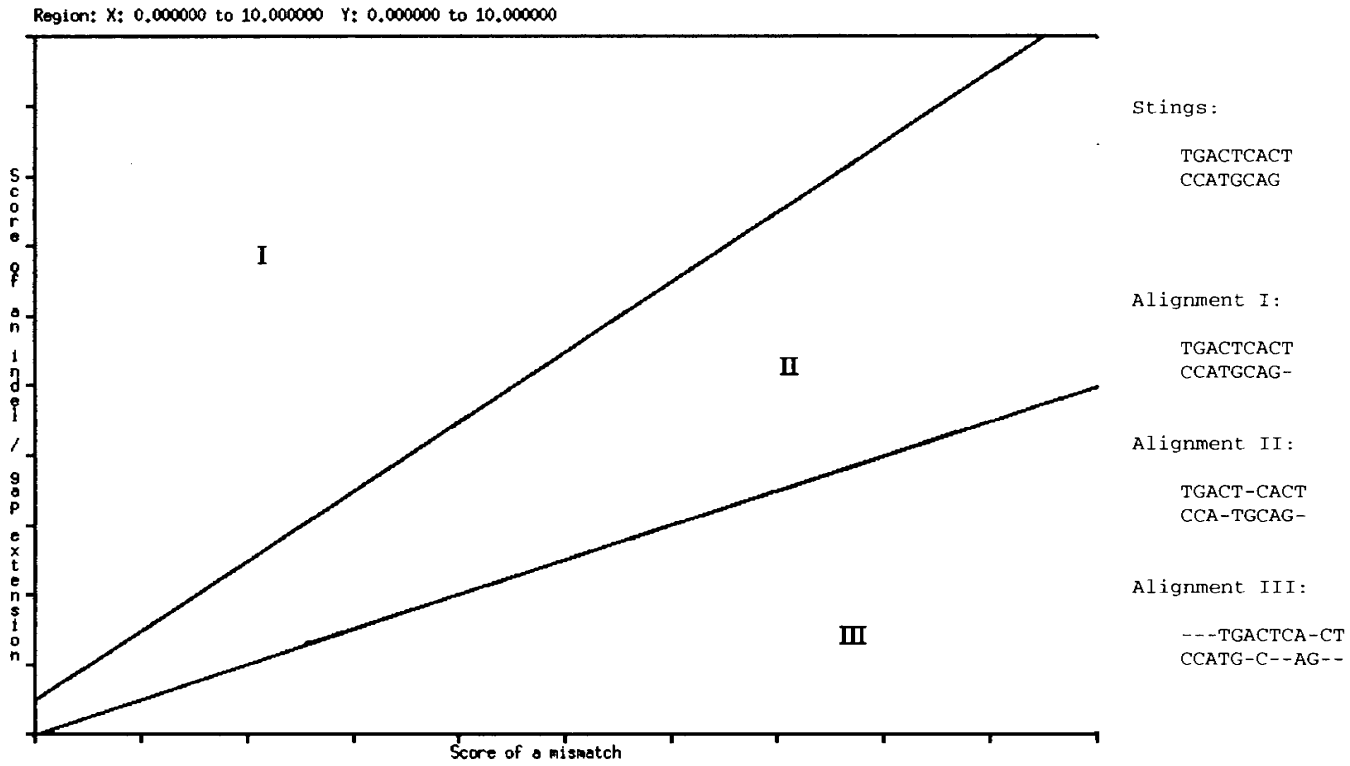

Figure 2.1: A polygonal decomposition of the α, β parameter space when $\gamma = 0$

Algorithm 1 Global alignment without gap penalties

$high = n$
$low = 0$
while $((high - low) > 1/n^2)$ **do**
  $mid = low + (high - low)/2$
  compute the optimal global alignment $\mathcal{A}(w, x, y)$ at point $(0, mid)$
  if $(y == y_0)$ **then**
    return the line passing through $(-1, -\frac{1}{2})$ and $(0, mid)$
  else
    if $(y > y_0)$ **then**
      $low = mid$
    else
      $high = mid$
    end if
  end if
end while
compute optimal global alignments $\mathcal{A}_{high}$, $\mathcal{A}_{low}$ for points $(0, high)$, $(0, low)$
$mid = low + (high - low)/2$
if ($\mathcal{A}_{high}$ is the same as $\mathcal{A}_{low}$) **then**
  return the line passing through $(-1, -\frac{1}{2})$ and $(0, mid)$
else
  compute β_0 such that $w_{high} - \beta_0 y_{high} = w_{low} - \beta_0 y_{low}$
  compute optimal alignment $\mathcal{A}$ for point $(0, \beta_0)$
  if $(w_0 - \beta_0 y_0 == w - \beta_0 y)$ **then**
    return the line passing through $(-1, -\frac{1}{2})$ and $(0, \beta_0)$
  else
    return the line passing through $(-1, -\frac{1}{2})$ and $(0, mid)$
  end if
end if

3. Global Alignment with Gap Penalty

In this chapter we solve the inverse parametric global alignment problem with gap penalties. There are now three parameters to consider, α, β and γ. Given a reference alignment $\mathcal{A}_0$ with w_0 matches, x_0 mismatches, y_0 indels and z_0 gaps, we need to find a point on the α, β, γ space where $\mathcal{A}_0$ is optimal or approximately optimal.

Preliminary Results

First, let us describe the boundary lines of the optimal regions in the α, β, γ space.

Theorem 3.1 (Gusfield et al. [7]) *Any line forming a boundary between three or more regions is of the form* $\beta = c + (c + 1/2)\alpha, \gamma = d + d\alpha$.

Corollary 3.1 *Suppose* $(\alpha_0, \beta_0, \gamma_0)$ *is an inverse-optimal point in the* α, β, γ *space for reference alignment* $\mathcal{A}_0$. *Then every point on the line that passes through* $(-1, -1/2, 0)$ *and* $(\alpha_0, \beta_0, \gamma_0)$ *is inverse-optimal for* $\mathcal{A}_0$.

Corollary 3.2 *All region boundaries intersect with either the positive* β, γ *coordinate plane or with the positive* α, γ *coordinate plane.*

As in Equation (2.1), the boundary line between optimal regions on the β, γ plane associated with alignments $\mathcal{A}_i$ and $\mathcal{A}_j$ has the form

$$\beta = \frac{w_i - w_j}{y_i - y_j} - \frac{z_i - z_j}{y_i - y_j}\gamma \tag{3.1}$$

A *vertex* is the intersection point of three or more optimal regions. Suppose vertex v is intersection point of three optimal regions whose optimal alignments are $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$. Let

$\Delta w_1 = w_1 - w_2, \Delta w_2 = w_2 - w_3$, etc. Then $v = (\beta_v, \gamma_v)$, where

$$\beta_v = \frac{\Delta w_1 \Delta z_2 - \Delta w_2 \Delta z_1}{\Delta y_1 \Delta z_2 - \Delta y_2 \Delta z_1} \quad \text{and} \quad \gamma_v = \frac{\Delta w_1 \Delta y_2 - \Delta w_2 \Delta y_1}{\Delta z_1 \Delta y_2 - \Delta z_2 \Delta y_1} \tag{3.2}$$

Finally, let us describe a property of centroids of convex polygons.

Theorem 3.2 (Grunbaum [5]) *Let S be a convex body of volume 1 in R^d. Let v_1 be the larger of the two volumes in a division of S by a hyperplane through its centroid. Then $v_1 \leq 1 - (\frac{d}{d+1})^d$.*

Description of the Algorithm

According to Corollary 3.2, if there is an inverse-optimal point on the search space, then there must be an inverse-optimal point on either the positive β, γ coordinate plane or the positive α, γ coordinate plane. Thus we can search on the β, γ ($\alpha = 0$) coordinate plane first. If an inverse-optimal point is found on the β, γ coordinate plane, algorithm terminates; otherwise, continue to search on the α, γ coordinate plane. If an inverse-optimal point is found on the α, γ coordinate plane, then return it. If there is no inverse-optimal point on the α, γ coordinate plane either, then return an approximate inverse-optimal point.

The algorithm uses the following idea to reduce the search space. Let $v = (0, \beta_v, \gamma_v)$ be a point in the remaining search space on the β, γ plane and let $\mathcal{A}_v$ be the optimal alignment at v, with w_v matches, x_v mismatches, y_v indels and z_v gaps. If reference alignment $\mathcal{A}_0$ is optimal at v, then v is an inverse-optimal point. Otherwise, suppose $\mathcal{A}_0$ is optimal at point $(0, \beta, \gamma)$. By the optimality of $\mathcal{A}_0$ and $\mathcal{A}_v$, it follows that:

$$w_0 - \beta y_0 - \gamma z_0 \geq w_v - \beta y_v - \gamma z_v \quad \text{and} \quad w_v - \beta_v y_v - \gamma_v z_v \geq w_0 - \beta_v y_0 - \gamma_v z_0.$$

Therefore,

$$(y_v - y_0)\beta + (z_v - z_0)\gamma \geq (y_v - y_0)\beta_v + (z_v - z_0)\gamma_v \tag{3.3}$$

If $\mathcal{A}_0$ is not an optimal alignment at any point on the β, γ plane, then suppose $(0, \beta, \gamma)$ is an approximately inverse-optimal point that minimizes the numerical difference between the score of the optimal alignment and the score of $\mathcal{A}_0$, and that $\hat{\mathcal{A}}$ is optimal at that point. Thus,

we have:

$$(w_v - \beta_v y_v - \gamma_v z_v) - (w_0 - \beta_v y_0 - \gamma_v z_0) \geq (\hat{w} - \beta\hat{y} - \gamma\hat{z}) - (w_0 - \beta y_0 - \gamma z_0)$$

$$\text{and} \qquad (\hat{w} - \beta\hat{y} - \gamma\hat{z}) \geq (w_v - \beta y_v - \gamma z_v)$$

Therefore

$$(y_v - y_0)\beta + (z_v - z_0)\gamma \geq (y_v - y_0)\beta_v + (z_v - z_0)\gamma_v,$$

which is the same as inequality (3.3).

The boundary line l of the halfplane defined by (3.3) passes through v, so l divides the remaining search space into two regions. The region whose points do not satisfy inequality (3.3) can be discarded, thereby reducing the search space. Line l becomes a new boundary line for the remaining search space; we say that this boundary line is *defined by* point v and alignment $\mathcal{A}_v$. According to the Theorem 3.2, if the *centroid* of the present search space is selected as point v, the area of search space will reduce by a factor of at least $4/9$.

Using the centroid as described above, we repeatedly reduce the search region on the β, γ plane until its area is smaller than $1/2m^7$. By Theorem 3.3, which is proved later, when the region is reduced to this size, it cannot include a complete optimal region.

After the binary search terminates, if there exists an inverse-optimal point, there must exist an inverse-optimal *vertex* in the remaining search space. From Lemma 3.2, proved later, the inverse-optimal vertex cannot occur on the boundary line of the remaining search space. From Theorem 3.4, also proved below, if there exists an inverse-optimal vertex (β^*, γ^*) in the remaining search space, there exist two distinct boundary lines l_1 and l_2, defined by (β_1, γ_1) and $\mathcal{A}_1$ and by (β_2, γ_2) and $\mathcal{A}_2$, such that $\mathcal{A}_1, \mathcal{A}_2$ that are optimal at (β^*, γ^*); that is, (β^*, γ^*) is the intersection of the scores of $\mathcal{A}_0$, $\mathcal{A}_1$, and $\mathcal{A}_2$, and we say (β^*, γ^*) is *located* (*determined*) by l_1 and l_2 or $\mathcal{A}_1$ and $\mathcal{A}_2$. From Lemma 3.4, if there exists an inverse-optimal vertex in the remaining search space, when m is big, we can use the two longest boundary lines to locate that vertex; when m is small enough, we need check different pairs of boundary lines to locate that inverse-optimal vertex.

The details of the algorithm are shown in Algorithm 2.

Algorithm Analysis

To show that Algorithm 2 is correct, we first need to prove some results.

Lemma 3.1 *The distance between any two vertices on the β, γ plane is greater than $1/m^3$ and the distance between a vertex and a boundary line of optimal regions is greater than $1/m^3$.*

Proof According to Equation (3.1) and (3.2), select a boundary line l of optimal regions and a vertex $v = (\beta_v, \gamma_v)$ as:

$$l : \beta = \frac{\Delta w_i}{\Delta y_i} - \frac{\Delta z_i}{\Delta y_i}\gamma \text{ and}$$

$$\beta_v = \frac{\Delta w_1 \Delta z_2 - \Delta w_2 \Delta z_1}{\Delta y_1 \Delta z_2 - \Delta y_2 \Delta z_1}, \gamma_v = \frac{\Delta w_1 \Delta y_2 - \Delta w_2 \Delta y_1}{\Delta z_1 \Delta y_2 - \Delta z_2 \Delta y_1}$$

Then the distance between v and l is:

$$d = \left| \beta_v + \frac{\Delta z_i}{\Delta y_i}\gamma_v - \frac{\Delta w_i}{\Delta y_i} \right| \cdot \frac{|\Delta y_i|}{\sqrt{\Delta y_i^2 + \Delta z_i^2}}$$

Since $-m < \Delta w_i, \Delta y_i, \Delta z_i < m$, when $d > 0$, $d > 1/m^3$

Since the distance between any two vertices should be greater than the distance between a vertex and a boundary line of an optimal region, the distance between any two vertices is greater than $1/m^3$. □

Theorem 3.3 *The area of any complete optimal region on the β, γ plane is greater than $1/2m^6$*

Proof A complete optimal region is composed of at least three vertices. Suppose the minimal complete optimal region is made with 3 vertices, then it is a triangle. By Lemmas 3.1, the base of this triangle has length greater than $1/m^3$ and the height of this triangle is greater than $1/m^3$. Therefore, the area is greater than $\frac{1}{2} \cdot \frac{1}{m^3} \cdot \frac{1}{m^3} = 1/2m^6$. □

The precondition for the following results is that the area of the remaining search space is smaller than $1/2m^7$.

Lemma 3.2 *The inverse-optimal vertex cannot be on the boundary of the remaining search space.*

Proof Assume that the reference alignment is $\mathcal{A}_0(w_0, y_0, z_0)$, that the inverse-optimal vertex (β^*, γ^*) is on a boundary line l of the remaining search space, and that the boundary line l is defined by (β_v, γ_v) and alignment $\mathcal{A}_v(w_v, y_v, z_v)$. Notice that $\mathcal{A}_0$ is not optimal at (β_v, γ_v), then $(\beta^*, \gamma^*) \neq (\beta_v, \gamma_v)$. Since (β^*, γ^*) is on l, according to Equation (3.3), we have:

$$(y_v - y_0)\beta^* + (z_v - z_0)\gamma^* = (y_v - y_0)\beta_v + (z_v - z_0)\gamma_v \qquad (*)$$

By the optimality of $\mathcal{A}_0$ and $\mathcal{A}_v$, we have:

$$w_0 - \beta^* y_0 - \gamma^* z_0 \geq w_v - \beta^* y_v - \gamma^* z_v \quad \text{and} \quad w_v - \beta_v y_v - \gamma_v z_v > w_0 - \beta_v y_0 - \gamma_v z_0.$$

Therefore,

$$(y_v - y_0)\beta^* + (z_v - z_0)\gamma^* > (y_v - y_0)\beta_v + (z_v - z_0)\gamma_v \qquad (**)$$

Equations (*), (**) are contradictory, so the assumption is wrong. Thus, the inverse-optimal vertex cannot be on any boundary line. □

Lemma 3.3 *Suppose reference alignment $\mathcal{A}_0(w_0, y_0, z_0)$ is optimal at vertex (β^*, γ^*) in the remaining search space. Let l be a boundary line of the remaining search space defined by (β_v, γ_v) and alignment $\mathcal{A}_v$. If $\mathcal{A}_v$ is not optimal at (β^*, γ^*), then the distance between (β^*, γ^*) and l is greater than $1/m^3$.*

Proof According to inequality (3.3), the boundary line l is

$$(y_v - y_0)\beta + (z_v - z_0)\gamma = (y_v - y_0)\beta_v + (z_v - z_0)\gamma_v \qquad (*)$$

Suppose $\mathcal{A}_v$ is not optimal at (β^*, γ^*). From the optimality of $\mathcal{A}_0$ and $\mathcal{A}_v$, we have:

$$w_0 - \beta^* y_0 - \gamma^* z_0 = w_v - \beta^* y_v - \gamma^* z_v + \Delta c_1, \quad \Delta c_1 > 0 \qquad (**)$$

$$w_v - \beta_v y_v - \gamma_v z_v = w_0 - \beta_v y_0 - \gamma_v z_0 + \Delta c_2, \quad \Delta c_2 > 0 \qquad (***)$$

Adding (**) and (***) we obtain

$$(y_v - y_0)\beta^* + (z_v - z_0)\gamma^* = (y_v - y_0)\beta_v + (z_v - z_0)\gamma_v + \Delta c_1 + \Delta c_2 \qquad (****)$$

Equations (*) and (****) define two parallel lines, and (****) passes through (β^*, γ^*). Thus the distance between (β^*, γ^*) and line l equals the distance between line (****) and line l. Hence the distance is:

$$d = \left| \frac{\Delta c_1 + \Delta c_2}{y_v - y_0} \cdot \frac{y_v - y_0}{\sqrt{(y_0 - y_v)^2 + (z_0 - z_v)^2}} \right| = \left| \frac{\Delta c_1 + \Delta c_2}{\sqrt{(y_0 - y_v)^2 + (z_0 - z_v)^2}} \right|$$

According to (**), $\Delta c_1 = w_0 - w_v - (y_0 - y_v)\beta^* - (z_0 - z_v)\gamma^* > 0$

Since (β^*, γ^*) is a vertex, according to Equation (3.2) it is given by an expression of the form:

$$\beta^* = \frac{\Delta w_1 \Delta z_2 - \Delta w_2 \Delta z_1}{\Delta y_1 \Delta z_2 - \Delta y_2 \Delta z_1} \quad \text{and} \quad \gamma^* = \frac{\Delta w_1 \Delta y_2 - \Delta w_2 \Delta y_1}{\Delta z_1 \Delta y_2 - \Delta z_2 \Delta y_1}$$

It is clear that $\Delta c_1 > 1/m^2$, then

$$d > \left| \frac{1}{m^2} \cdot \frac{1}{\sqrt{(y_0 - y_v)^2 + (z_0 - z_v)^2}} \right| > \frac{1}{m^3}$$

□

Theorem 3.4 *If there exists an inverse-optimal vertex (β^*, γ^*) in the remaining search space, then there exist two boundary lines that are defined by (β_1, γ_1), where alignment $\mathcal{A}_1$ is optimal, and (β_2, γ_2), where alignment $\mathcal{A}_2$ is optimal, and alignments $\mathcal{A}_1, \mathcal{A}_2$ are optimal at (β^*, γ^*).*

Proof Suppose reference alignment $\mathcal{A}_0$ is optimal at vertex (β^*, γ^*) in the remaining search space.

Assume that no boundary line that is defined by (β_i, γ_i), alignment $\mathcal{A}_i$ and $\mathcal{A}_i$ is optimal at (β^*, γ^*). Then from Lemma 3.3, the distance between (β^*, γ^*) and any point on the boundary of the remaining search space is greater than $1/m^3$. Hence the area of the remaining search space is greater than $1/m^6$. But the area of the remaining search space is smaller than $1/2m^7$, a contradiction. So there exists at least one boundary line that is defined by (β_1, γ_1), alignment $\mathcal{A}_1$ and $\mathcal{A}_1$ is optimal at (β^*, γ^*).

If there is only one boundary line that satisfies the above requirements, we can find a contradiction from similar reasoning. Thus there exist at least two boundary lines that satisfy above requirements. □

Lemma 3.4 *If there exists an inverse-optimal vertex in the remaining search space, then when* $m > 80$, *it can be located using the two longest boundary lines.*

Proof Suppose that there exists an inverse-optimal *vertex* in the remaining search space. Notice that if the distance of the two farthest points on the boundary is smaller than $1/m^3$, then any two boundary lines can be used to locate the inverse-optimal *vertex* (according to Lemma 3.3).

Since the search space can be initially restricted to $0 \leq \beta \leq m^2, 0 \leq \gamma \leq m^2$, the area of remaining search space is smaller than $1/2m^7$, and every iteration reduces the search by at least $4/9$ and increases the number of boundary lines by at most one, there are at most $\log_{\frac{9}{5}} 2 + 11 \log_{\frac{9}{5}} m$ boundary lines of the remaining search space.

If the length of a boundary line l is more than $1/m^4$, then l can be used to locate that inverse-optimal *vertex*. Thus if the lengths of the two longest boundary lines are greater than $1/m^4$, then the inverse-optimal *vertex* can be located by them. If there are no two boundary lines whose lengths are greater than $1/m^4$, then the distance of the two farthest points on the boundary is smaller than $d = (\log_{\frac{9}{5}} 2 + 11 \log_{\frac{9}{5}} m) \cdot \frac{1}{m^4}$. When $m > 80$, $d < \frac{1}{m^3}$, thus the two longest boundary lines can be used to locate the inverse-optimal *vertex* too. □

Now we prove the correctness of Algorithm 2 and analyze its running time.

Theorem 3.5 *If there exists an inverse-optimal point on the* β, γ *coordinate plane, then Algorithm 2 can find it in* $O(mn \log m)$ *time.*

Proof According to Equation 3.2, the maximum coordinate for a vertex is (m^2, m^2). Thus, we can restrict the search space to $0 \leq \beta \leq m^2, 0 \leq \gamma \leq m^2$. If the algorithm finds an inverse-optimal point (β_v, γ_v) during the binary search, it returns a line. Theorem 3.3 shows

that when the area of the remaining search space is smaller than $1/2m^7$, it cannot include a complete optimal region. At this time, if there exist inverses an optimal point, it can not occur in a complete optimal region; it can only occur at a single *vertex* or single boundary line of the optimal regions. When the inverse optimal points occur on the boundary line of the optimal regions, there are two inverse optimal *vertices*. So at this time binary search terminates and the algorithm begins to check the vertices in the remaining region. From Lemma 3.2 the inverse-optimal vertex cannot occur on the boundary. If there exists an inverse-optimal vertex in the remaining search space, from Theorem 3.4 and Lemma 3.4, steps 11-32 can find it. So if there exists inverse-optimal point on β, γ coordinate plane, Algorithm 2 can find that point.

The initial area of search space is m^4; the binary search terminates when the area is less than $1/2m^7$. Every iteration reduces the area of search space by at least 4/9 and increases the number of boundary lines by at most 1, so there are $O(\log m)$ iterations and $O(\log m)$ boundary lines. In every iteration, the algorithm computes one optimal alignment which takes $O(mn)$ time. Thus steps 1-10 need $O(mn \log m)$ time. Step 11 needs $O(\log m)$ time, since there are $O(\log m)$ intersection points on the boundary and the two farthest points must both be intersection points. Steps 12-13 need $O(\log m)$ time, since there are $O(\log m)$ lines on the boundary of the remaining search space. Function *checkvertex* needs $O(mn)$ time, then steps 15-16 needs $O(mn)$ time. In step 17-32, since $m < 80$, we can consider the number of different pairs of boundary lines as constant, then the time needed is $O(mn)$. Thus the total time need by Algorithm 2 is $O(mn \log m)$. □

Algorithm 2 also works for searching on the α, γ plane, only that, in step 35, when we have found that the reference alignment cannot be optimal on the α, γ coordinate plane, we need to return the centroid of the remaining search space as an approximately inverse-optimal point.

Algorithm 2 Global alignment with gap penalties

set the search space $R = \{(0, \beta, \gamma) | 0 \leq \beta \leq m^2, 0 \leq \gamma \leq m^2\}$
while $(Area(R) > 1/2m^7)$ **do**
  compute the centroid $v(\beta_v, \gamma_v)$ of search space R
  compute the optimal global alignment $\mathcal{A}_v$ at point v on β, γ space
  if ($\mathcal{A}_v$ is the same as $\mathcal{A}_0$) **then**
    return the line passing through $(-1, -\frac{1}{2}, 0)$ and $(0, \beta_v, \gamma_v)$
  else
    $R \leftarrow R \cap$ the halfplane defined by Equation (3.3)
  end if
end while
$d = max\{|uv| \big|$ u,v are points on the boundary of R$\}$
$l_1 =$ the length of the longest boundary line of R
$l_2 =$ the length of the second longest boundary line of R
if ($d < 1/m^3$ or $l_2 \geq 1/m^4$ or $m > 80$) **then**
  alignments $\mathcal{A}_1, \mathcal{A}_2$ define the boundary line l_1, l_2
  checkvertex$(\mathcal{A}_1, \mathcal{A}_2, R)$
else
  if ($l_1 \geq 1/m^4$) **then**
    store all boundary lines of R except l_1 into stack S
    alignment $\mathcal{A}_1$ defines the boundary line l_1
    while (S is not empty) **do**
      pop a boundary line defined by alignment $\mathcal{A}_2$ from S
      checkvertex$(\mathcal{A}_1, \mathcal{A}_2, R)$
    end while
  else
    store all pairs of different boundary lines of R into stack S
    while (S is not empty) **do**
      pop a pair of boundary lines defined by alignments $\mathcal{A}_1, \mathcal{A}_2$ from S
      checkvertex$(\mathcal{A}_1, \mathcal{A}_2, R)$
    end while
  end if
end if
continue search on α, γ plane

Function 1 checkvertex$(\mathcal{A}_1, \mathcal{A}_2, R)$

compute $(\hat{\beta}, \hat{\gamma})$ so $\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2$ have same score at $(\hat{\beta}, \hat{\gamma})$
if ($(\hat{\beta}, \hat{\gamma})$ is in R) **then**
  compute the optimal global alignment $\hat{A}$ at point $(\hat{\beta}, \hat{\gamma})$
  if ($\hat{\mathcal{A}}$ and $\mathcal{A}_0$ have same score at $(\hat{\beta}, \hat{\gamma})$) **then**
    return the line passing through $(-1, -\frac{1}{2}, 0)$ and $(0, \hat{\beta}, \hat{\gamma})$
  end if
end if

4. Further Results and Open Problems

In this Chapter, we give the algorithm for inverse parametric local alignment without gaps problem and discuss the futures work.

The inverse local alignment problem without gaps ($\gamma = 0$) asks to find an inverse-optimal or approximately inverse-optimal point on the α, β coordinate plane. This problem can be solved by slightly modifying Algorithm 2 of Chapter 3.

Note that in this case, we search over the α, β coordinate plane. Similar to equations (3.1),(3.2),(3.3), the boundary line between optimal regions on the α, β plane has the form

$$\alpha = \frac{w_i - w_j}{x_i - x_j} - \frac{y_i - y_j}{x_i - x_j}\beta, \tag{4.1}$$

vertex v on the α, β plane has the form

$$\alpha_v = \frac{\Delta w_1 \Delta y_2 - \Delta w_2 \Delta y_1}{\Delta x_1 \Delta y_2 - \Delta x_2 \Delta y_1} \quad \text{and} \quad \beta_v = \frac{\Delta w_1 \Delta x_2 - \Delta w_2 \Delta x_1}{\Delta y_1 \Delta x_2 - \Delta y_2 \Delta x_1} \tag{4.2}$$

and the inequality used to reduce the search space has the form

$$(x_v - x_0)\alpha + (y_v - y_0)\beta \geq (x_v - x_0)\alpha_v + (y_v - y_0)\beta_v \tag{4.3}$$

Lemmas 3.1, 3.2, 3.3, 3.4 and Theorems 3.3, 3.4 all hold for the α, β plane. Thus we can modify Algorithm 2 as follows: search on the α, β plane instead of the β, γ plane; compute the local alignment instead of global alignment. Details are given in Algorithm 3. The algorithm analysis in Chapter 3 implies that this algorithm runs in $O(mn \log m)$ time.

An open problem is to extend our binary search strategy into fixed-dimensional space. For example, this could lead to an efficient algorithm for inverse local alignment with gaps, which is a search problem in the three-dimensional α, β, γ space.

Algorithm 3 Local alignment without gap penalties

set the search space $R = \{(\alpha, \beta) | 0 \leq \alpha \leq m^2, 0 \leq \beta \leq m^2\}$
while $(Area(R) > 1/2m^7)$ **do**
  compute the centroid $v(\alpha_v, \beta_v)$ of search space R
  compute the optimal local alignment $\mathcal{A}_v$ at point v on α, β space
  if ($\mathcal{A}_v$ is the same as $\mathcal{A}_0$) **then**
    return the point (α_v, β_v)
  else
    $R \leftarrow R \cap$ the halfplane defined by Equation (4.3)
  end if
end while
$d = max\{|uv| \big| \text{u,v are points on the boundary of R}\}$
$l_1 =$ the length of the longest boundary line of R
$l_2 =$ the length of the second longest boundary line of R
if ($d < 1/m^3$ or $l_2 \geq 1/m^4$ or $m > 80$) **then**
  alignments $\mathcal{A}_1, \mathcal{A}_2$ define the boundary line l_1, l_2
  checkvertex$(\mathcal{A}_1, \mathcal{A}_2, R)$
else
  if ($l_1 \geq 1/m^4$) **then**
    store all boundary lines of R except l_1 into stack S
    alignment $\mathcal{A}_1$ defines the boundary line l_1
    while (S is not empty) **do**
      pop a boundary line defined by alignment $\mathcal{A}_2$ from S
      checkvertex$(\mathcal{A}_1, \mathcal{A}_2, R)$
    end while
  else
    store all pairs of different boundary lines of R into stack S
    while (S is not empty) **do**
      pop a pair of boundary lines defined by alignments $\mathcal{A}_1, \mathcal{A}_2$ from S
      checkvertex$(\mathcal{A}_1, \mathcal{A}_2, R)$
    end while
  end if
end if
return the centroid of the remaining search space R

Function 2 checkvertex$(\mathcal{A}_1, \mathcal{A}_2, R)$

compute $(\hat{\alpha}, \hat{\beta})$ so $\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2$ have same score at $(\hat{\alpha}, \hat{\beta})$
if ($(\hat{\alpha}, \hat{\beta})$ is in R) **then**
  compute the optimal local alignment $\hat{A}$ at point $(\hat{\alpha}, \hat{\beta})$
  if ($\hat{A}$ and $\mathcal{A}_0$ have same score at $(\hat{\alpha}, \hat{\beta})$) **then**
    return the point $(\hat{\alpha}, \hat{\beta})$
  end if
end if

Acknowledgments

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this thesis. First and foremost, Dr. David Fernández-Baca for his guidance, patience and support throughout this research and the writing of this thesis. His insights and words of encouragement have often inspired me and renewed my hopes for completing my graduate education. I would also like to thank my committee members for their efforts and contributions to this work: Dr. Soma Chaudhuri, Dr. Srinivas Aluru and Dr. Govindarasu Manimaran. I would additionally like to thank Dr. Wei Yu for his help throughout my research.

Bibliography

[1] A. Apostolico and R. Giancarlo. Sequence alignment in molecular biology. *Journal of Computational Biology*, 5(2):173–196, 1998.

[2] D. Eppstein. Setting parameters by example. *Proc. 40th Symp. Foundations of Computer Science, IEEE*, pages 309–318, 1999.

[3] D. Fernández-Baca, T. Seppäläinen, and G. Slutzki. Bounds for parametric sequence comparison. *Discrete Applied Mathematics*, 2002, to appear.

[4] W. Fitch and T. F. Smith. Optimal sequence alignments. *Proceedings of the National Academy of Sciences of the USA*, 80:1382–1386, 1983.

[5] B. Grunbaum. Partitions of mass distributions and of convex bodies by hyperplanes. *Pacific Journal of Mathematics*, 10:1257–1261, 1960.

[6] D. Gusfield. Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge University Press, Cambridge, New York, Melbourne, 1997.

[7] D. Gusfield, K. Balasubramanian, and D. Naor. Parametric optimization of sequence alignment. *Algorithmica*, 12:312–326, 1994.

[8] D. Gusfield and P. Stelling. Parametric and inverse-parametric sequence alignment with XPARAL. *Methods in Enzymology*, 226:481–494, 1996.

[9] N. Megiddo. Combinatorial optimization with rational objective functions. *Mathematics of Operations Research*, 4:414–424, 1979.

[10] N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *Journal of the ACM*, 30(4):852–865, 1983.

[11] D. Sankoff and E. J. Kruskal. Time Warps, String Edits, and Macromolecules: the theory and practice of sequence comparison. *Addison-Wesley*, 1983.

[12] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[13] M. Vingron and M. Waterman. Sequence alignment and penalty choice: review of concepts, case studies, and implications. *Journal of Molecular Biology*, 235:1–12, 1994.

[14] M. S. Waterman. Parametric and ensemble sequence alignment. *Bulletin of Mathematical Biology*, 56(4):743–767, 1994.

[15] M. S. Waterman, M. Eggert, and E. Lander. Parametric sequence comparisons. *Proceedings of the National Academy of Sciences of the USA*, 89:6090–6093, 1992.